Advanced Encryption Standard (AES)

Objectives

- To review a short history of AES
- To define the basic structure of AES
- To define the transformations used by AES
- To define the key expansion process
- To discuss different implementations

7-1 INTRODUCTION

The Advanced Encryption Standard (AES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST) in December 2001.

Topics discussed in this section:

7.1.1 History
7.1.2 Criteria
7.1.3 Rounds
7.1.4 Data Units
7.1.5 Structure of Each Round

7.1.2 Criteria

The criteria defined by NIST for selecting AES fall into three areas:

1. Security

2. Cost

3. Implementation.

7.1.3 Rounds.

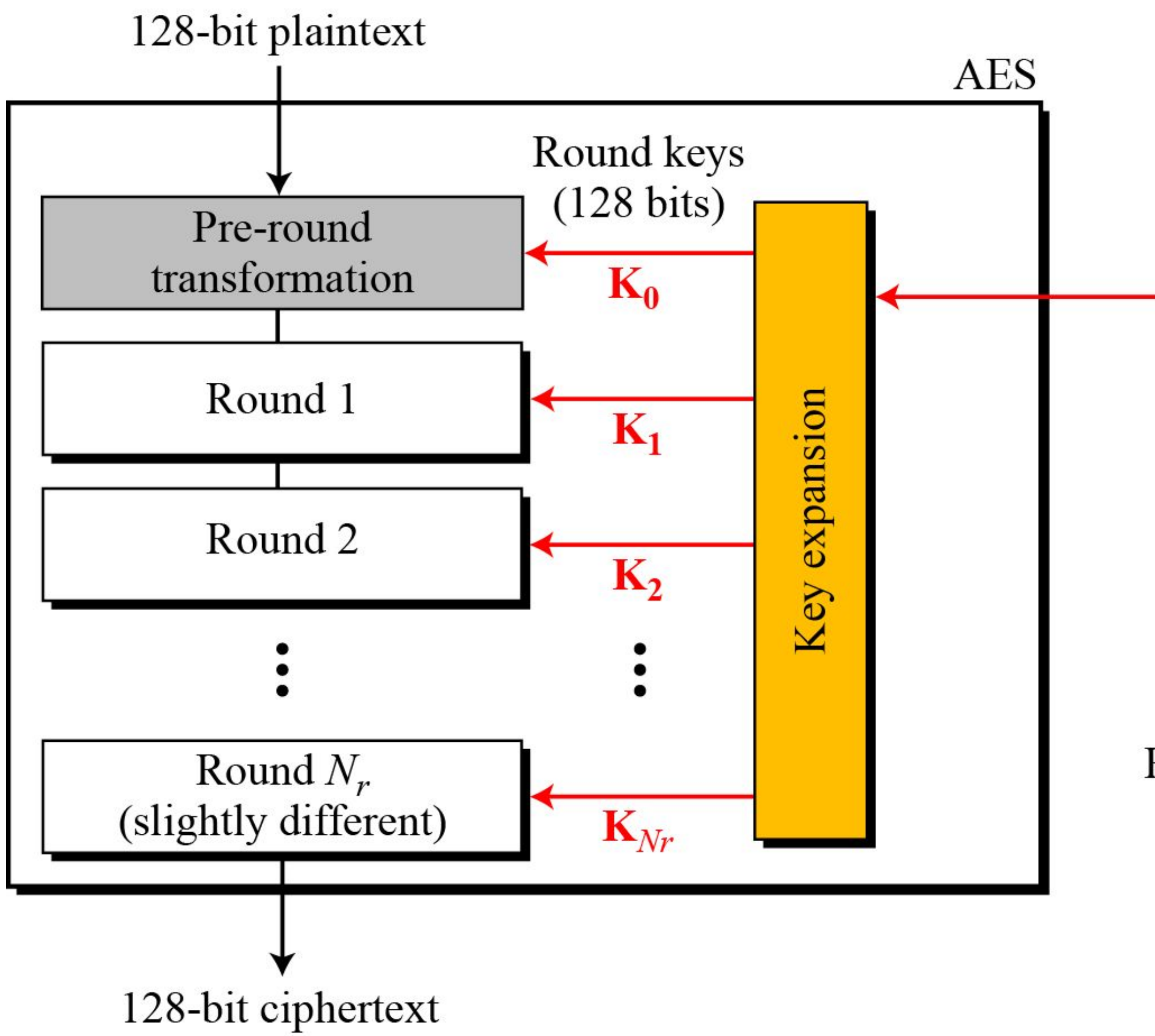
AES is a non-Feistel cipher that encrypts and decrypts a data block of 128 bits. It uses 10, 12, or 14 rounds. The key size, which can be 128, 192, or 256 bits, depends on the number of rounds.

Note

AES has defined three versions, with 10, 12, and 14 rounds.
Each version uses a different cipher key size (128, 192, or 256), but the round keys are always 128 bits.

7.1.3 Continue

Figure 7.1 *General design of AES encryption cipher*

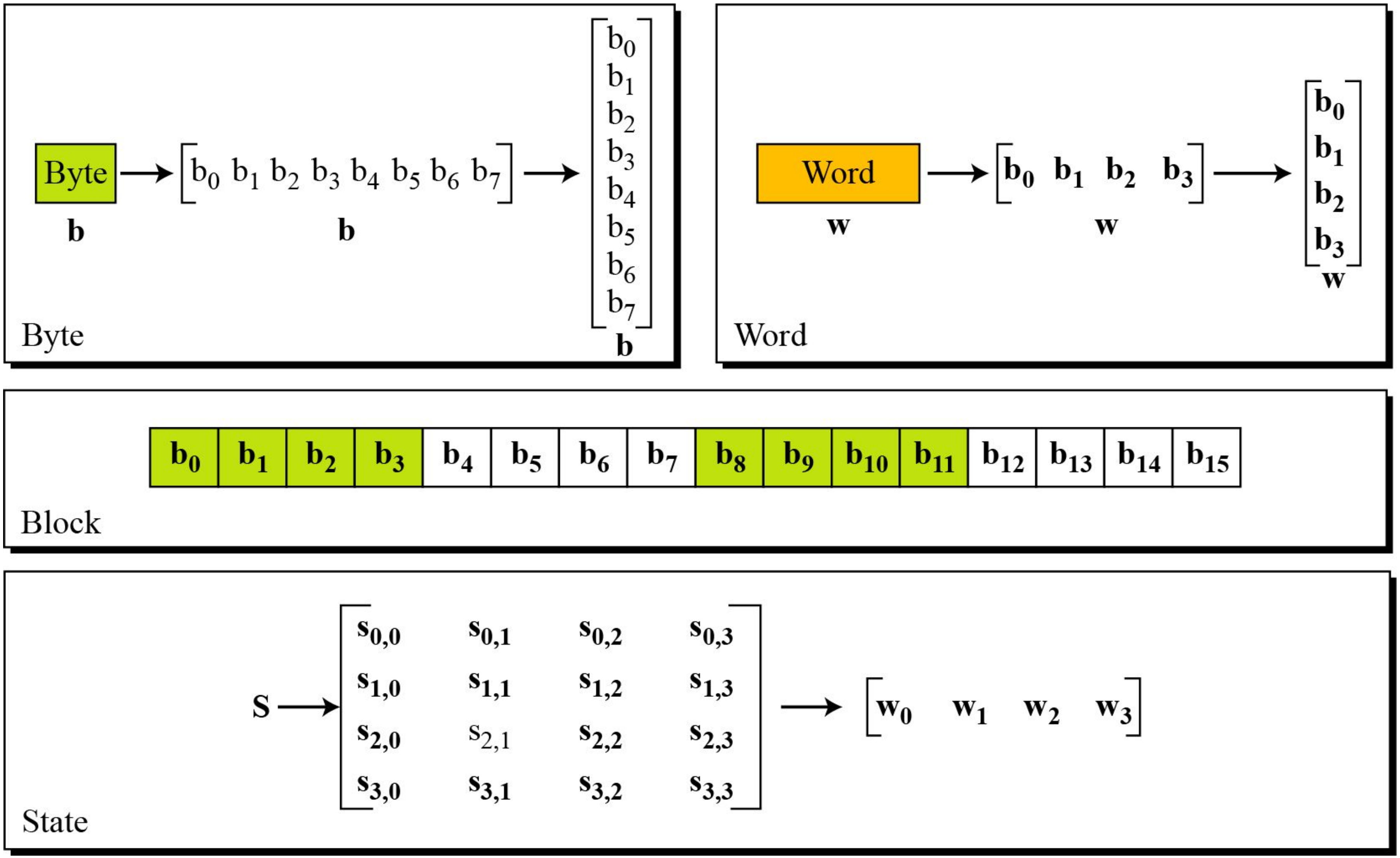


Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

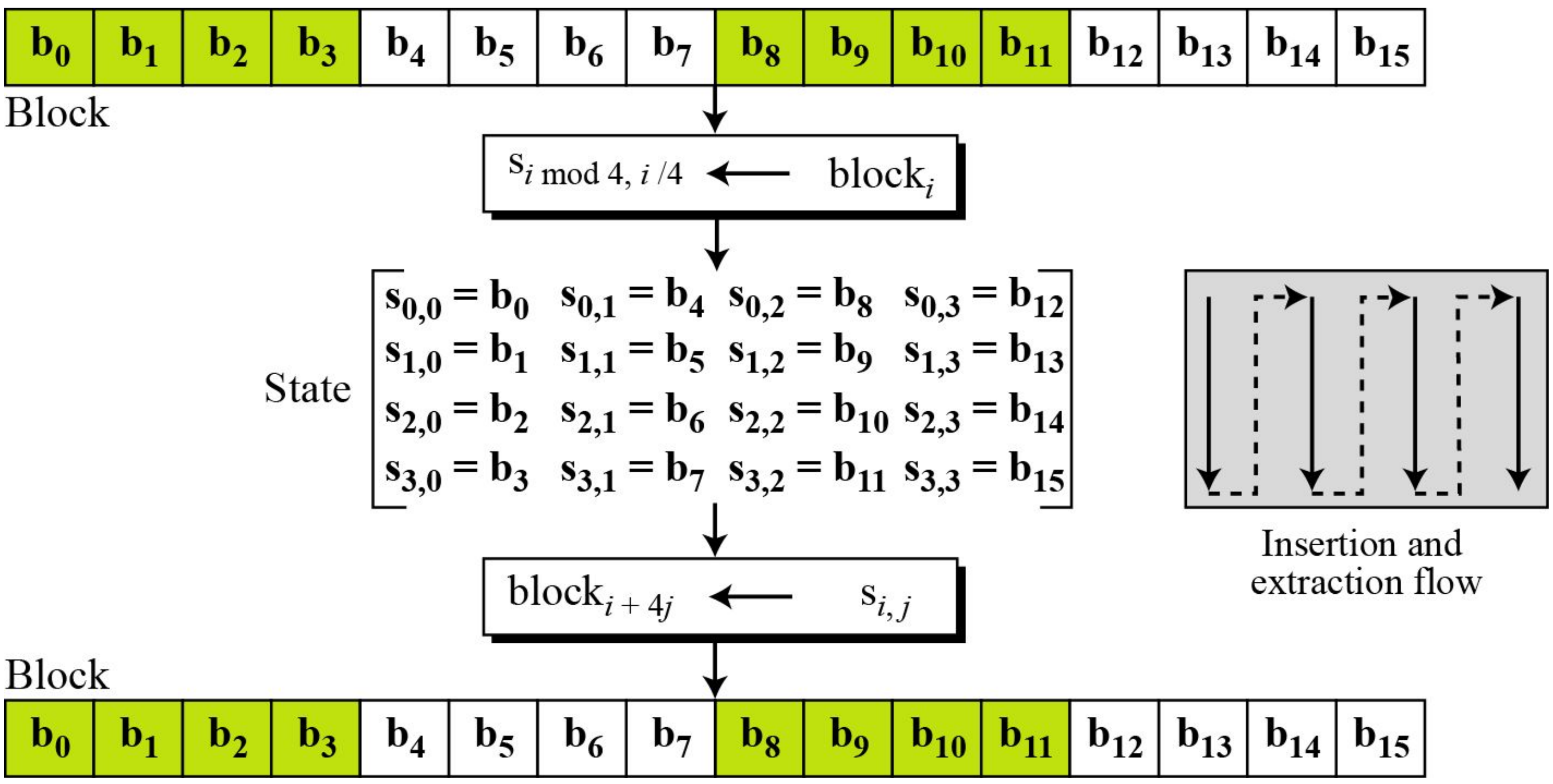
7.1.4 Data Units.

Figure 7.2 *Data units used in AES*



7.1.4 Continue

Figure 7.3 *Block-to-state and state-to-block transformation*



7.1.4 *Continue*

Example 7.1 *Continue*

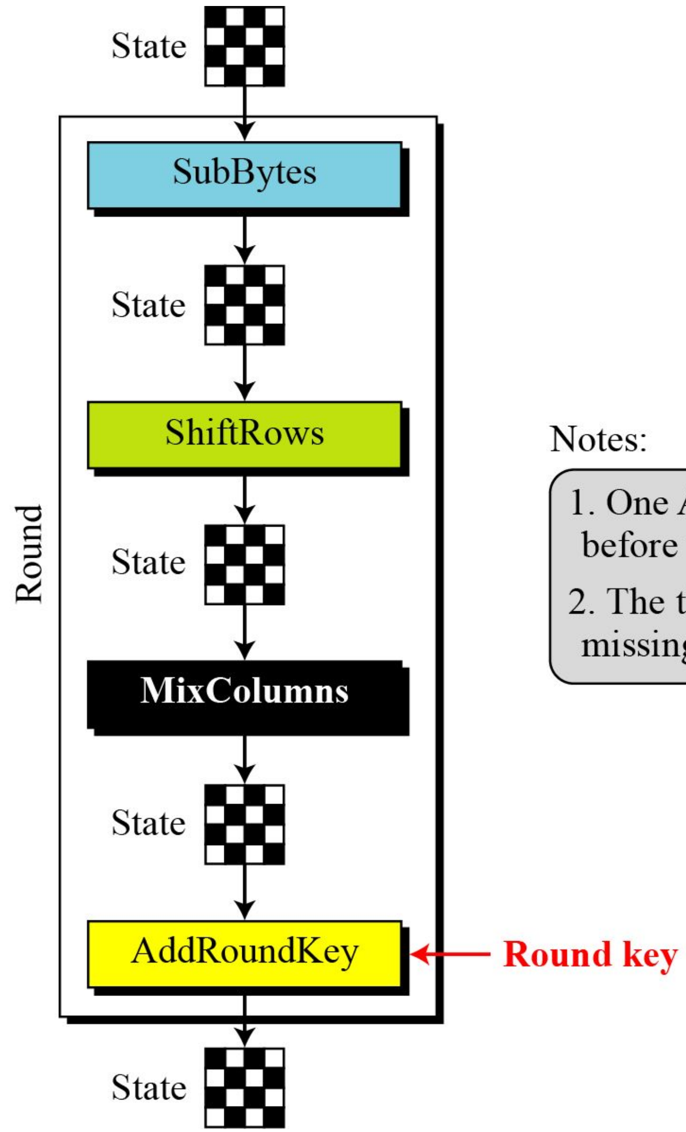
Figure 7.4 ***Changing plaintext to state***

Text	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Hexadecimal	00	04	12	14	12	04	12	00	0C	00	13	11	08	23	19	19

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix} \text{State}$$

7.1.5 Structure of Each Round

Figure 7.5 *Structure of each round at the encryption site*



Notes:

1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.

7-2 TRANSFORMATIONS

To provide security, AES uses four types of transformations: substitution, permutation, mixing, and key-adding.

<u>Topics discussed in this section:</u>

7.2.1 Substitution
7.2.2 Permutation
7.2.3 Mixing
7.2.4 Key Adding

7.2.1 Substitution

AES, like DES, uses substitution. AES uses two invertible transformations.

SubBytes

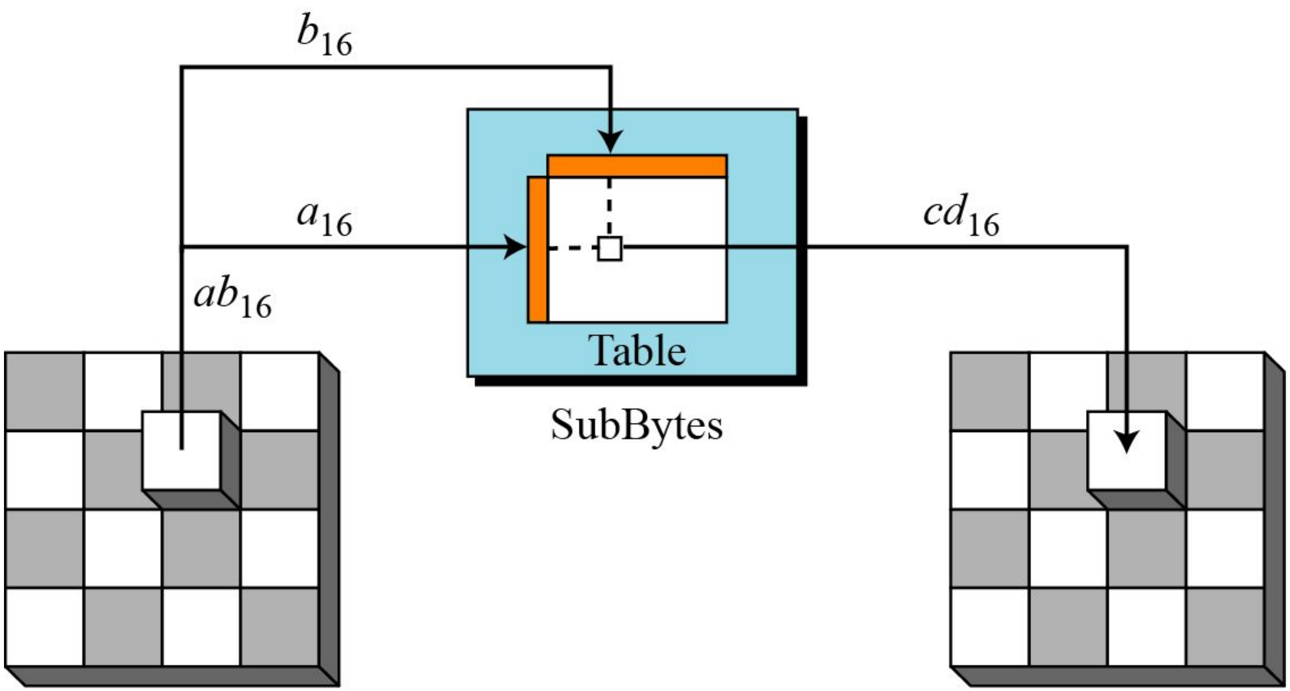
The first transformation, SubBytes, is used at the encryption site. To substitute a byte, we interpret the byte as two hexadecimal digits.

Note

The SubBytes operation involves 16 independent byte-to-byte transformations.

7.2.1 Continue

Figure 7.6 ***SubBytes transformation***



7.2.1 Continue

Table 7.1 *SubBytes transformation table*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8

7.2.1 Continue

Table 7.1 *SubBytes transformation table (continued)*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	CB	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

InvSubBytes

Table 7.2 *InvSubBytes transformation table*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B

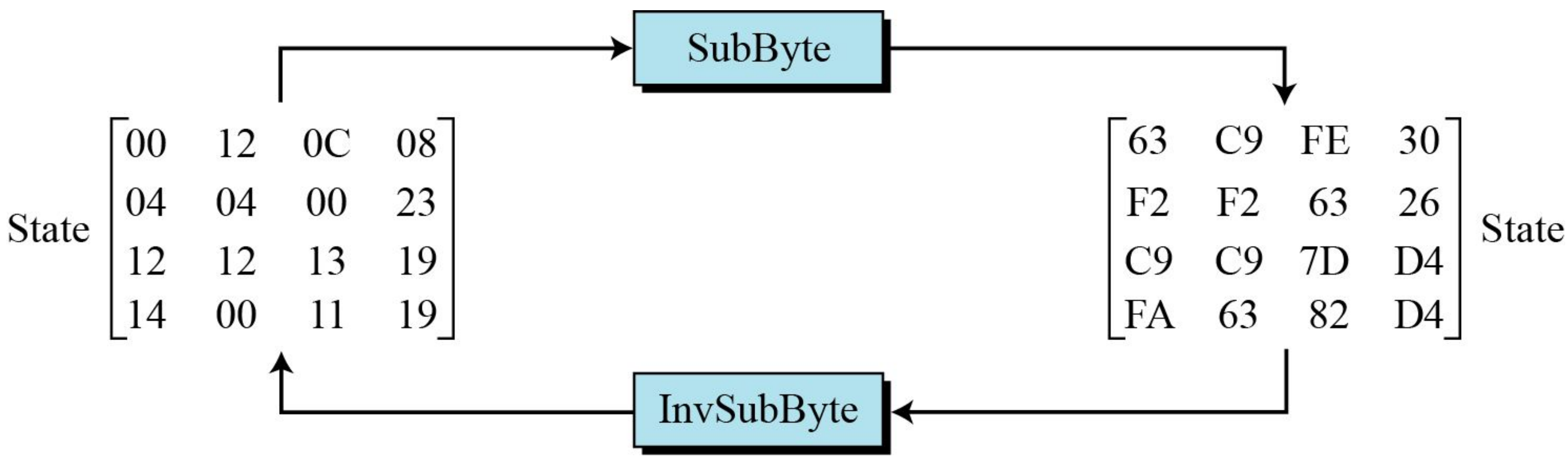
InvSubBytes (Continued)

8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Example 7.2

Figure 7.7 shows how a state is transformed using the SubBytes transformation. The figure also shows that the InvSubBytes transformation creates the original one. Note that if the two bytes have the same values, their transformation is also the same.

Figure 7.7 *SubBytes transformation for Example 7.2*



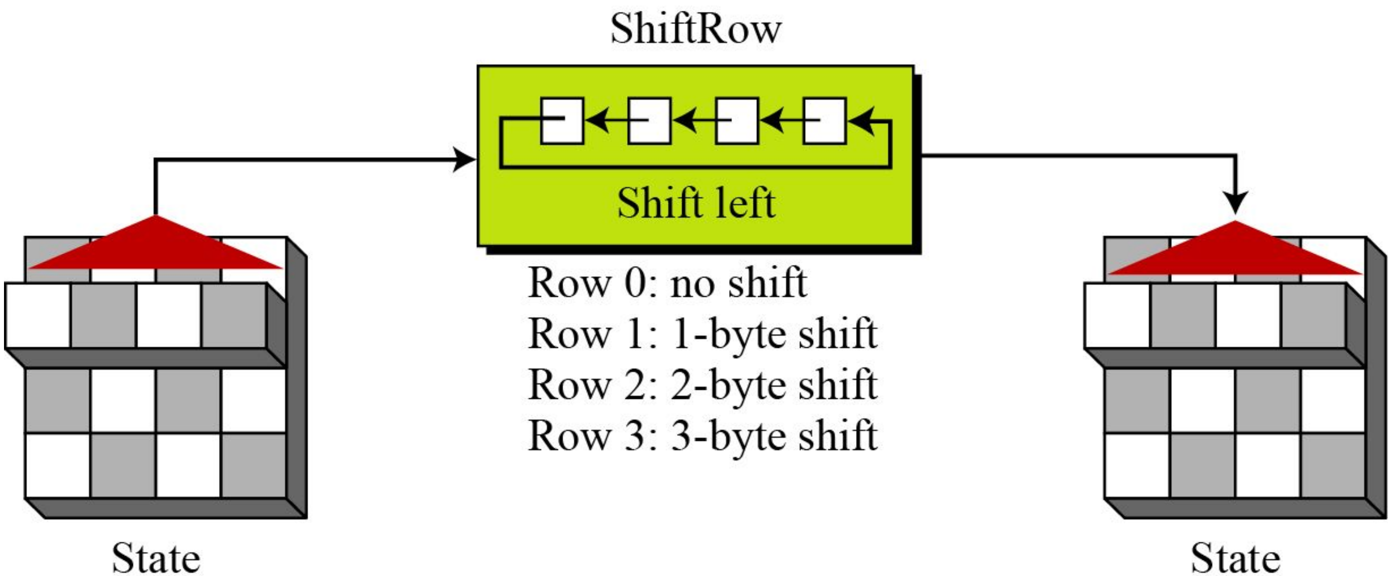
7.2.2 Permutation

Another transformation found in a round is shifting, which permutes the bytes.

ShiftRows

In the encryption, the transformation is called ShiftRows.

Figure 7.9 ***ShiftRows transformation***



7.2.2 Continue

InvShiftRows

In the decryption, the transformation is called InvShiftRows and the shifting is to the right.

Algorithm 7.2 *Pseudocode for ShiftRows transformation*

```
ShiftRows (S)
{
    for (r = 1 to 3)
        shiftrow (s_r, r)                // s_r is the rth row
}
```

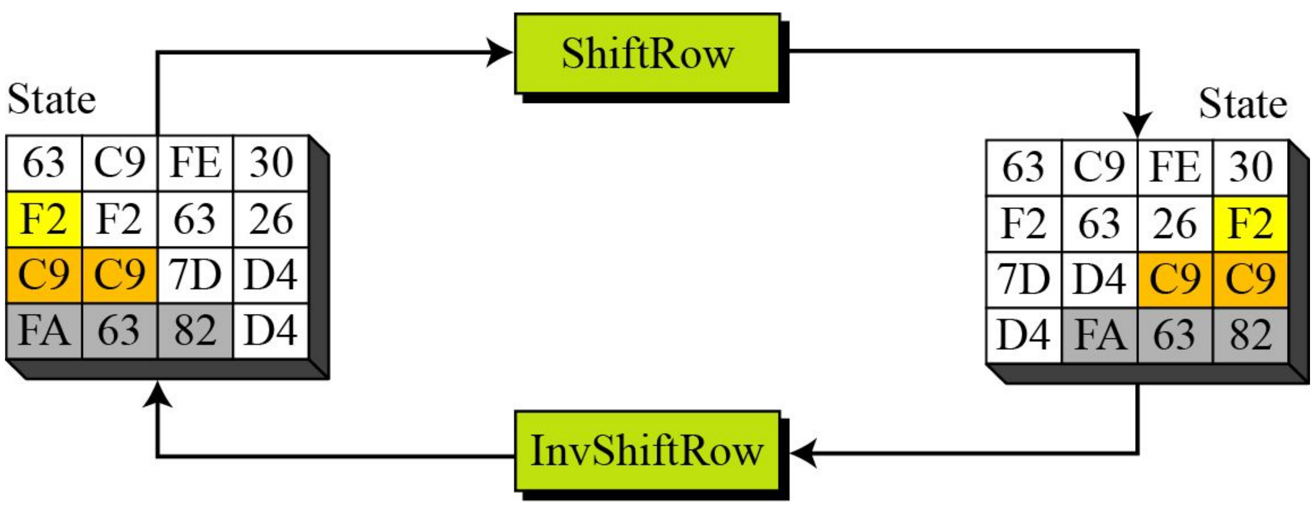
```
shiftrow (row, n)                       // n is the number of bytes to be shifted
{
   CopyRow (row, t)                     // t is a temporary row
   for (c = 0 to 3)
        row_((c − n) mod 4) ← t_c
}
```

7.2.2 Continue

Example 7.4

Figure 7.10 shows how a state is transformed using ShiftRows transformation. The figure also shows that InvShiftRows transformation creates the original state.

Figure 7.10 ***ShiftRows transformation in Example 7.4***



7.2.3 Mixing

We need an interbyte transformation that changes the bits inside a byte, based on the bits inside the neighboring bytes. We need to mix bytes to provide diffusion at the bit level.

Figure 7.11 *Mixing bytes using matrix multiplication*

$$\begin{matrix} a\mathbf{x} + b\mathbf{y} + c\mathbf{z} + d\mathbf{t} \\ e\mathbf{x} + f\mathbf{y} + g\mathbf{z} + h\mathbf{t} \\ i\mathbf{x} + j\mathbf{y} + k\mathbf{z} + l\mathbf{t} \\ m\mathbf{x} + n\mathbf{y} + o\mathbf{z} + p\mathbf{t} \end{matrix} \begin{matrix} \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \end{matrix} \begin{bmatrix} \square \\ \square \\ \square \\ \square \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \times \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \\ \mathbf{t} \end{bmatrix}$$

New matrix | **Constant matrix** | Old matrix

7.2.3 Continue

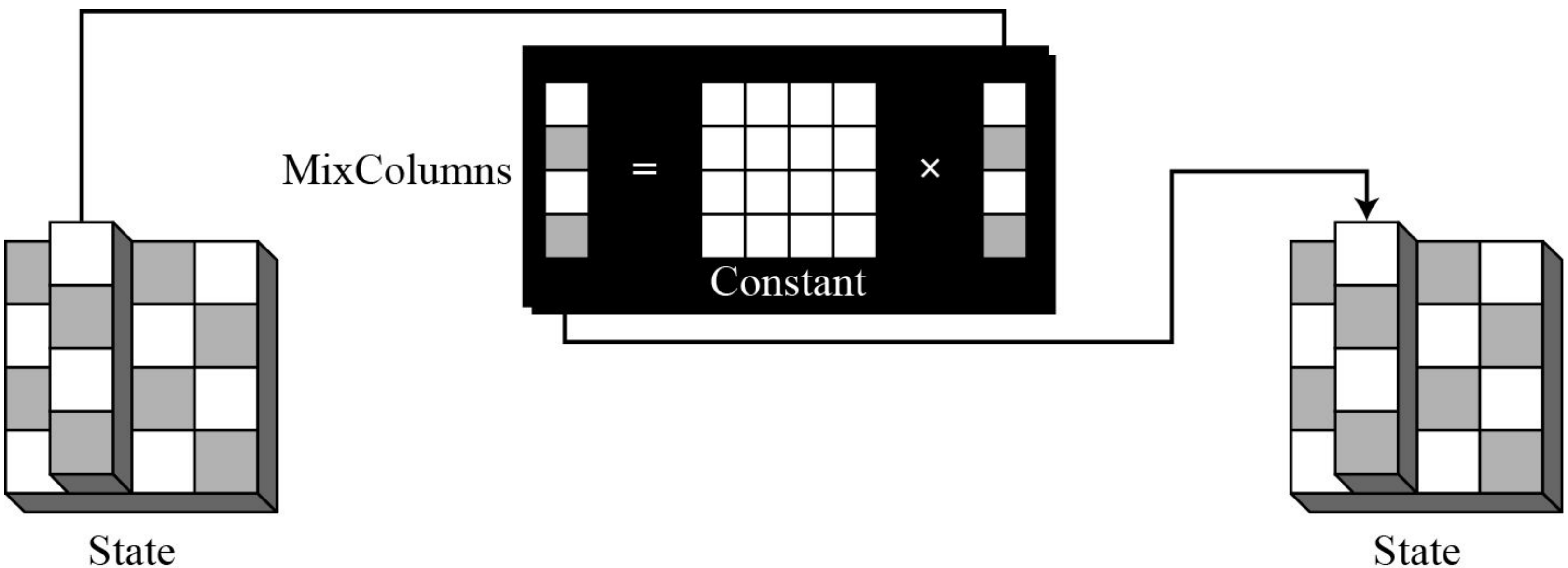
Figure 7.12 ***Constant matrices used by MixColumns and InvMixColumns***

$$\underset{C}{\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}} \xleftrightarrow{\text{Inverse}} \underset{C^{-1}}{\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}}$$

MixColumns

The MixColumns transformation operates at the column level; it transforms each column of the state to a new column.

Figure 7.13 *MixColumns transformation*



7.2.3 Continue

InvMixColumns

The InvMixColumns transformation is basically the same as the MixColumns transformation.

Note

The MixColumns and InvMixColumns transformations are inverses of each other.

Example 7.5

Figure 7.14 shows how a state is transformed using the MixColumns transformation. The figure also shows that the InvMixColumns transformation creates the original one.

Figure 7.14 *The MixColumns transformation in Example 7.5*

$$\text{State}\ \begin{bmatrix} 63 & C9 & FE & 30 \\ F2 & 63 & 26 & F2 \\ 7D & D4 & C9 & C9 \\ D4 & FA & 63 & 82 \end{bmatrix} \underset{\textbf{InvMixColumn}}{\overset{\textbf{MixColumn}}{\rightleftarrows}} \begin{bmatrix} 62 & 02 & 27 & 26 \\ CF & 92 & 91 & 0D \\ 0C & 0C & F4 & D6 \\ 99 & 18 & 30 & 74 \end{bmatrix}\ \text{State}$$

7.2.4 Key Adding

AddRoundKey

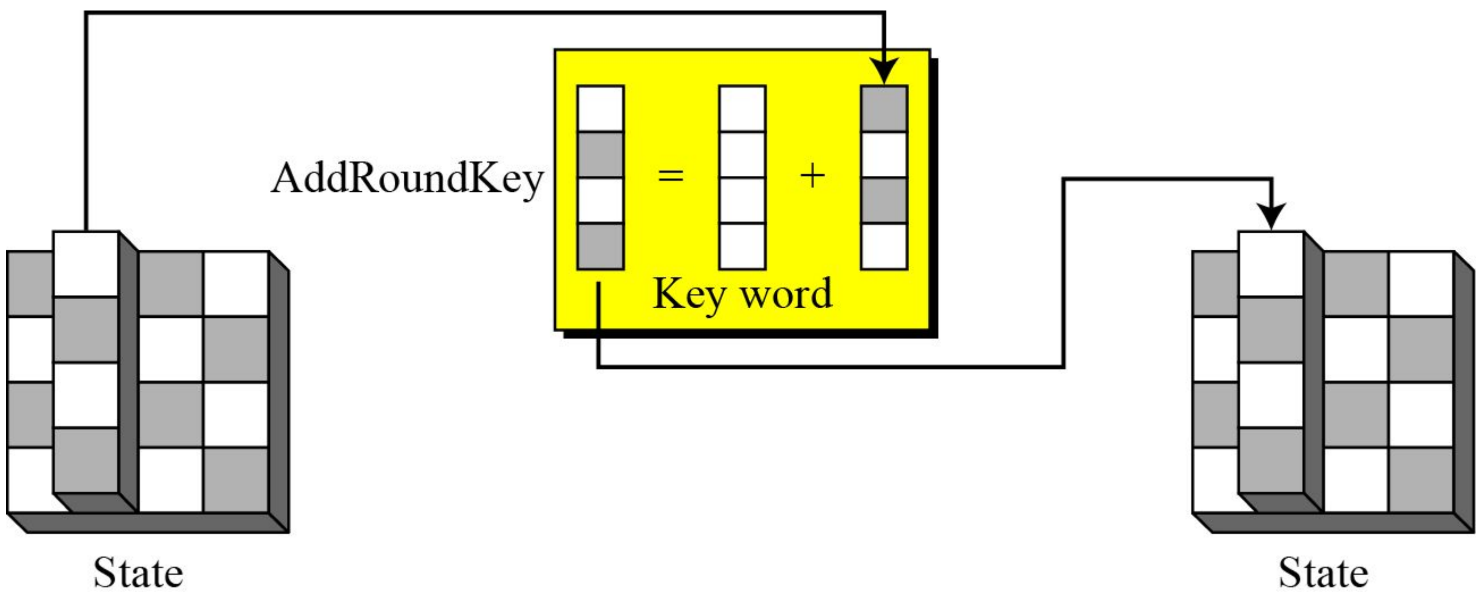
AddRoundKey proceeds one column at a time. AddRoundKey adds a round key word with each state column matrix; the operation in AddRoundKey is matrix addition.

Note

The AddRoundKey transformation is the inverse of itself.

7.2.4 Continue

Figure 7.15 ***AddRoundKey transformation***



Algorithm 7.4 *Pseudocode for AddRoundKey transformation*

AddRoundKey (S)
{
for (c = 0 to 3)
$\mathbf{s}_c \leftarrow \mathbf{s}_c \oplus \mathbf{w}_{\text{round} + 4c}$
}

7-3 KEY EXPANSION

To create round keys for each round, AES uses a key-expansion process. If the number of rounds is N_r, the key-expansion routine creates $N_r + 1$ 128-bit round keys from one single 128-bit cipher key.

Topics discussed in this section:

7.3.1 Key Expansion in AES-128
7.3.2 Key Expansion in AES-192 and AES-256
7.3.3 Key-Expansion Analysis

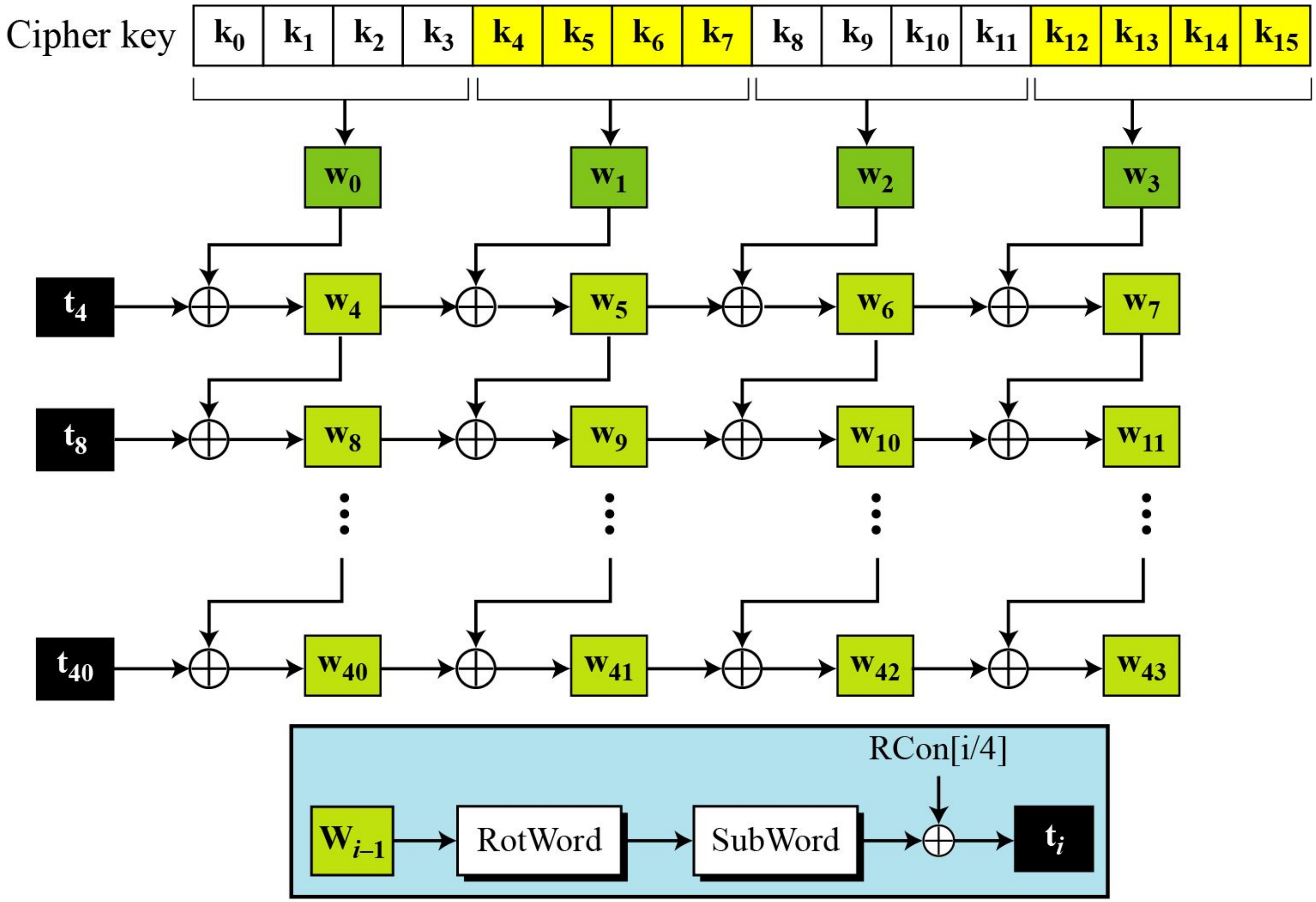
7-3 Continued

Table 7.3 *Words for each round*

Round	*Words*			
Pre-round	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$
1	$\mathbf{w}_4$	$\mathbf{w}_5$	$\mathbf{w}_6$	$\mathbf{w}_7$
2	$\mathbf{w}_8$	$\mathbf{w}_9$	$\mathbf{w}_{10}$	$\mathbf{w}_{11}$
. . .	. . .			
N_r	$\mathbf{w}_{4N_r}$	$\mathbf{w}_{4N_r+1}$	$\mathbf{w}_{4N_r+2}$	$\mathbf{w}_{4N_r+3}$

7.3.1 Key Expansion in AES-128

Figure 7.16 *Key expansion in AES*



Making of t_i (temporary) words $i = 4\ N_r$

7.3.1 Continue

Table 7.4 *RCon constants*

Round	*Constant (RCon)*	*Round*	*Constant (RCon)*
1	$(\underline{\mathbf{01}}\ 00\ 00\ 00)_{16}$	6	$(\underline{\mathbf{20}}\ 00\ 00\ 00)_{16}$
2	$(\underline{\mathbf{02}}\ 00\ 00\ 00)_{16}$	7	$(\underline{\mathbf{40}}\ 00\ 00\ 00)_{16}$
3	$(\underline{\mathbf{04}}\ 00\ 00\ 00)_{16}$	8	$(\underline{\mathbf{80}}\ 00\ 00\ 00)_{16}$
4	$(\underline{\mathbf{08}}\ 00\ 00\ 00)_{16}$	9	$(\underline{\mathbf{1B}}\ 00\ 00\ 00)_{16}$
5	$(\underline{\mathbf{10}}\ 00\ 00\ 00)_{16}$	10	$(\underline{\mathbf{36}}\ 00\ 00\ 00)_{16}$

7.3.2 Key Expansion in AES-192 and AES-256

Key-expansion algorithms in the AES-192 and AES-256 versions are very similar to the key expansion algorithm in AES-128, with the following differences:

7.3.3 Key-Expansion Analysis

The key-expansion mechanism in AES has been designed to provide several features that thwart the cryptanalyst.

7-4 CIPHERS

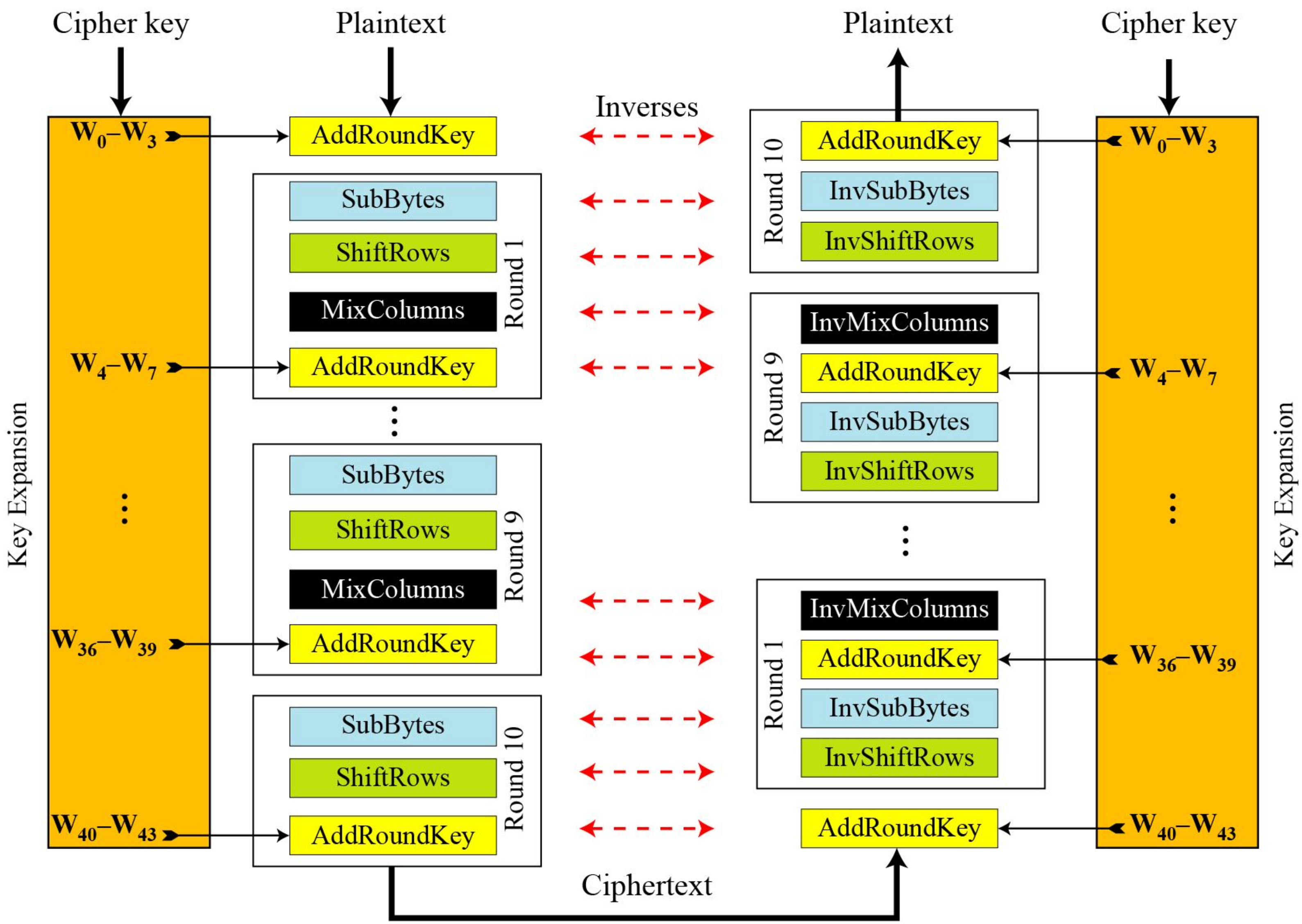
AES uses four types of transformations for encryption and decryption. In the standard, the encryption algorithm is referred to as the cipher and the decryption algorithm as the inverse cipher.

Topics discussed in this section:

7.4.1 Original Design
7.4.2 Alternative Design

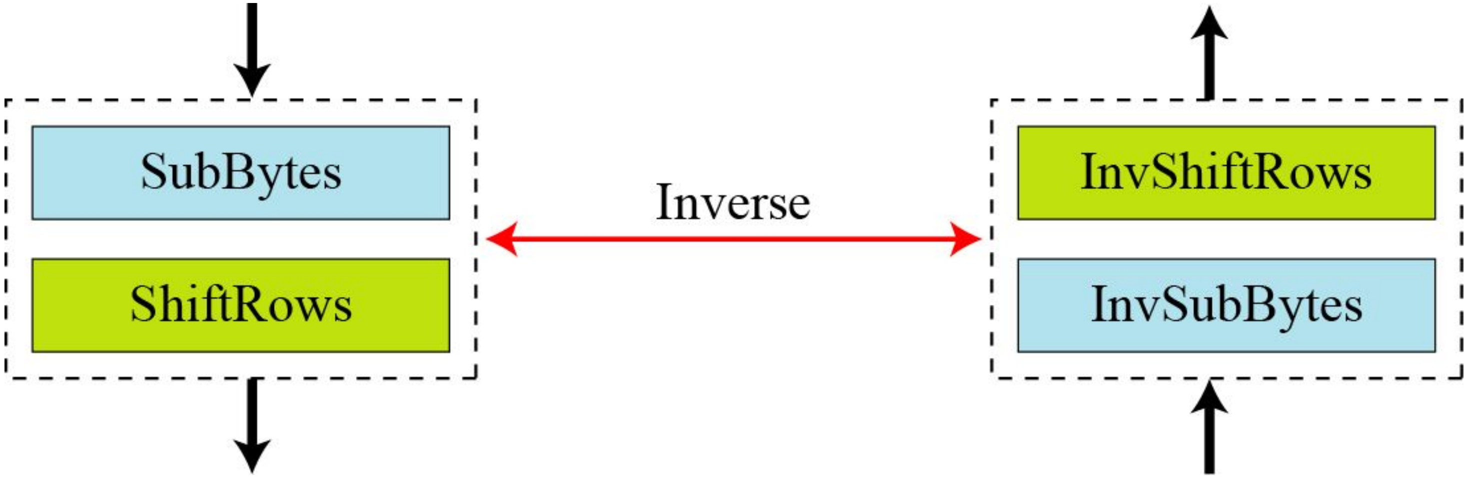
7.4.1 Original Design

Figure 7.17 *Ciphers and inverse ciphers of the original design*



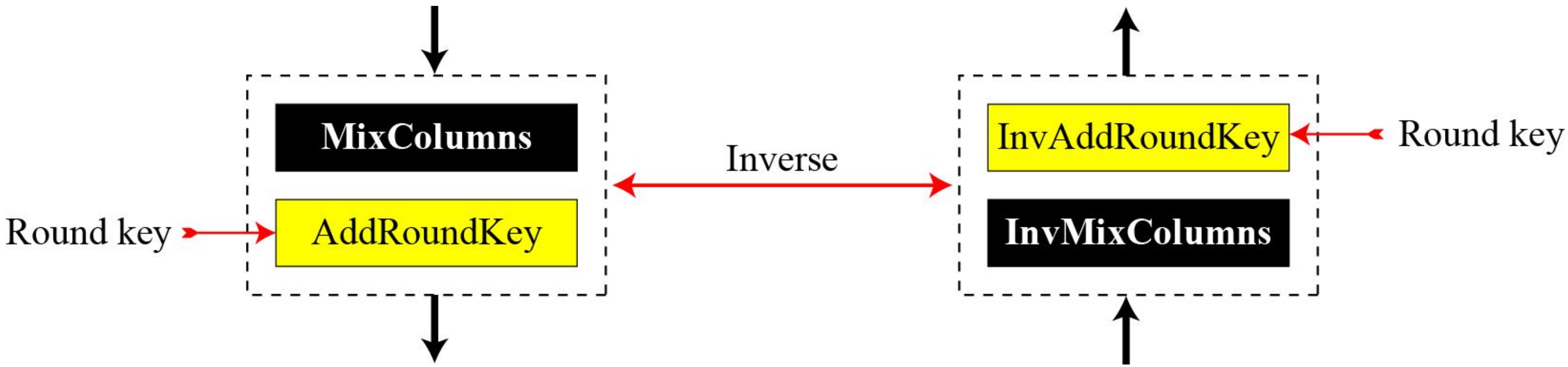
7.4.2 Alternative Design

Figure 7.18 ***Invertibility of SubBytes and ShiftRows combinations***



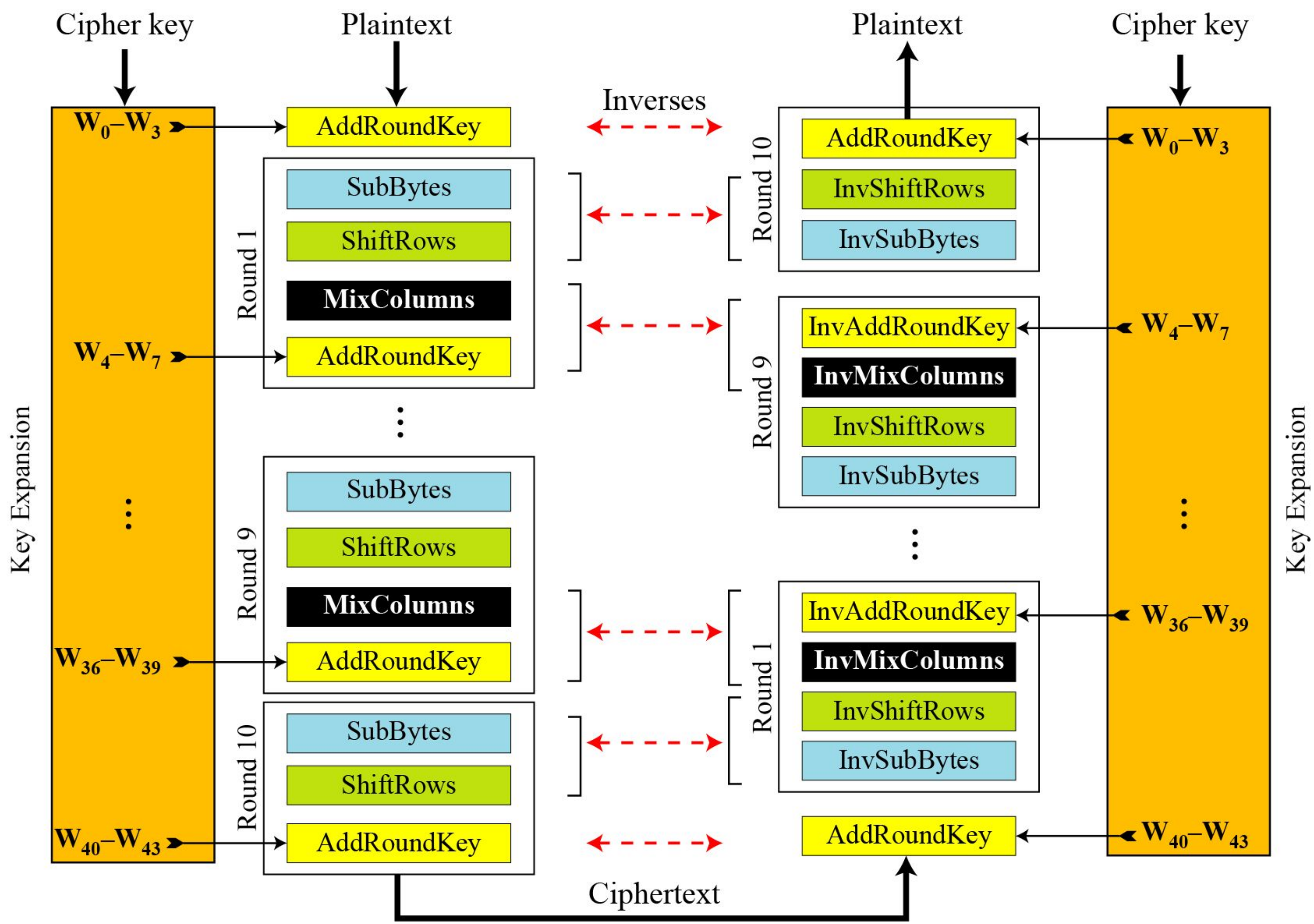
7.4.2 Continue

Figure 7.19 *Invertibility of MixColumns and AddRoundKey combination*



7.4.2 Continue

Figure 7.20 *Cipher and reverse cipher in alternate design*



7-6 ANALYSIS OF AES

This section is a brief review of the three characteristics of AES.

Topics discussed in this section:

7.6.1 Security
7.6.2 Implementation
7.6.3 Simplicity and Cost

7.6.1 Security

AES was designed after DES. Most of the known attacks on DES were already tested on AES.

Brute-Force Attack

AES is definitely more secure than DES due to the larger-size key.

Statistical Attacks

Numerous tests have failed to do statistical analysis of the ciphertext.

Differential and Linear Attacks

There are no differential and linear attacks on AES as yet.

7.6.2 Implementation

AES can be implemented in software, hardware, and firmware.